REMARKS

Claims 1-63 are pending in this application. By this paper, the Applicant amends claims 1, 22, and 43, and cancels claims 2, 4, 23, 25, 44 and 46 without prejudice. No new matter has been added.

At paragraph 5 of the most recent Office Action (dated July 28, 2008), the Examiner rejects claims 1, 22 and 43 under 35 U.S.C 102(b) as being anticipated by Muth, "ALTO: A Platform for Object Code Modification," 1999 (hereinafter Muth).

At paragraph 6 of the most recent Office Action, the Examiner rejects claims 1-63 under 35 U.S.C 102(b) as being anticipated by Hsu, "A Robust Foundation for Binary Translation of X86 Code," 1997 (hereinafter Hsu).

In light of the amendments and arguments set forth below, the Applicant believes all pending claims should be allowable.

Analysis of Hsu's Translator

The Hsu translator performs translations when a segment of a binary is loaded (page 21). Some segments are loaded when the program is started (PRELOAD segments, page 22). The Hsu translator uses control-flow analysis to explore each segment from its entry points, translating each block that it encounters (page 25). Having discovered the control flow of a segment, the entire segment is translated (Figure 4.5 page 30).

Sometimes, the control-flow analysis will not find all executable code (page 26), requiring "incremental" translation to occur. (See also page 35). Each segment has the original code at the entry points overwritten with breakpoint instructions (page 31) and when these breakpoints are hit, the segment can be translated (page 32). After a code segment **A** is edited (i.e. the code shuffled around), the translator iterates through all other code segments, looking to fix-up any hard-coded jumps to segment **A** that have moved (page 38).

Callback functions (where the address of a function call is passed to the operating system, to be called later) are handled by placing trap instructions at the start of all published function entry points, so when the OS invokes a function, the trap is triggered (page 39). If a section was pre-translated before the program started, this does not need to be done, because the address of the translated code would have been passed to the OS - this is only a problem when an address might have been passed to the OS prior to translation.

Calls to targets whose address cannot be resolved at translation time are replaced with calls to a "VMM function" that performs a lookup and transfers control (page 45-46). If the target has not been translated, it occurs there and then (page 51,54).

The Applicant has carefully considered the Response to Arguments and Objections raised in most recent Office Action.

At the second paragraph of page 3 of that Office Action, the Examiner states that section 6.1 of Hsu describes subject code containing self-modifying code being translated and allocated into memory. This not the case. This section of Hsu uses the term "translation" in relation to a description of a x86 segment addressing model. The address translation described in this section of Hsu explains how an x86 processor manages the physical memory addresses in RAM with respect to logical address space that is available to and used by a programmer. This address translation performed by x86 processors when running a program is not equivalent to the dynamic binary translation of the present invention.

In the Hsu rejection, the Examiner states that the self-modifying code detector as claimed is anticipated by the description at Section 6.2 of Hsu. This is not the case. Reading on to page 58, Hsu states that "*The detection strategy used in the translator consists of comparing the information of the LDT* (Local Descriptor-Table) *entries that are obtained before and after the user program is executed*". The detection strategy of Hsu therefore does not involve identifying self-modifying code during translation.

There are no checks in relation to self-modifying code performed during translation in the system described by Hsu. Hsu explicitly states on page 59 that "*This thesis's policy, however, is to*

minimize the risk of translating the self-modifying code". When the system described in Hsu is performing a translation it does not, indeed cannot identify self modifying code. It is for this reason that Hsu sets out to minimize the risk of translating self-modifying code, thereby avoiding the problems associated with translations of self-modifying code.

The Examiner states that page 26, lines 1-3 of Hsu describe the merging of instructions to form a larger single area, in a way that is relevant to the identification of self-modifying code as claimed. This is not the case, as in Hsu the process of amalgamating code is only applied to code which is not self-modifying. This is explained on page 24 of Hsu - the translator operates to merge instructions when a binary is loaded, and not when the translation is actually performed.

To more clearly distinguish the subject matter of the independent claims from the teachings of Hsu, the Applicant makes a number of amendments. First, by amending the introductory portion and final paragraph (c) of the independent claims, it is clear that claims as amended now relate to generating translated target code, and that the steps set out as (a) and (b) are performed as part of the process of generating translated target code.

The points of novelty of step (a) have been covered above.

Following the present amendments, step (b) sets out the structure and contents of the subject instruction groups - structures and contents that are novel over the teaching of Hsu. The benefits of the step as now claimed are in relation to generating a translation of self-modifying subject code in such a way to allow efficient management of the execution of the translation. Generating a translation of self-modifying code is something that Hsu deliberately sets out to avoid.

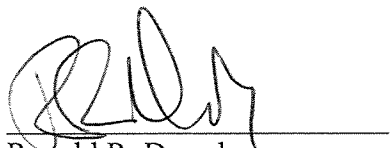
The amendments to the independent claims explained above likewise introduce features that are not taught or suggested in Muth.

In view of the above amendment, the Applicant believes the pending application is in condition for allowance. If the Examiner does not agree, the Applicant asks the Examiner to contact the undersigned to arrange a telephone interview to discuss the application, cited art and potential amendments.

Please charge our Deposit Account No. 08-0219, under Order No. 1801270.00139US1 from which the undersigned is authorized to draw.

Respectfully submitted,

Dated: February 27, 2009



Ronald R. Demsher
Registration No.: 42,478
Attorney for Applicant(s)

Wilmer Cutler Pickering Hale and Dorr LLP
60 State Street
Boston, Massachusetts 02109
(617) 526-6000 (telephone)
(617) 526-5000 (facsimile)